

1 CLAIMS

2
3 1. One or more computer-readable media having stored thereon a
4 plurality of instructions that, when executed by one or more processors of a
5 computer, causes the one or more processors to perform acts including:

6 allowing operation of the computer to begin based on untrusted code;
7 loading, under control of the untrusted code, a trusted core into memory;
8 preventing each of one or more central processing units and each of one or
9 more bus masters in the computer from accessing the memory;
10 resetting each of the one or more central processing units;
11 allowing one central processing unit to access the memory and execute
12 trusted core initialization code to initialize the trusted core; and
13 after execution of the trusted core has been initialized, allowing any other
14 central processing units and any bus masters in the computer to access the
15 memory.

16
17 2. One or more computer-readable media as recited in claim 1, wherein
18 the one or more processors comprise one or more controllers of one or more
19 memory controllers.

20
21 3. One or more computer-readable media as recited in claim 2, wherein
22 the one or more memory controllers are distributed among the one or more central
23 processing units.
24
25

1 4. One or more computer-readable media as recited in claim 2, wherein
2 the plurality of instructions comprise microcode to be executed by the one or more
3 memory controllers.

4
5 5. One or more computer-readable media as recited in claim 1, wherein
6 the untrusted code includes code from a basic input output system (BIOS) and
7 code from a plurality of option read only memories (ROMs).

8
9 6. One or more computer-readable media as recited in claim 1, wherein
10 the preventing comprises preventing each of the one or more central processing
11 units and each of the one or more bus masters from accessing the memory in
12 response to an initialize trusted core command received from one of the one or
13 more central processing units.

14
15 7. One or more computer-readable media as recited in claim 1, wherein
16 the loading the trusted core comprises copying different portions of the trusted
17 core from a plurality of different sources.

18
19 8. One or more computer-readable media as recited in claim 1, wherein
20 the loading the trusted core comprises copying different parts of the trusted core
21 from one or more sources and combining the different parts to assemble the
22 trusted core.

1 9. One or more computer-readable media as recited in claim 1, wherein
2 combining the different parts comprises exclusive-ORing bits of the different
3 parts.

4
5 10. One or more computer-readable media as recited in claim 1, wherein
6 the loading the trusted core comprises copying at least a portion of the trusted core
7 from a local mass storage device into the memory.

8
9 11. One or more computer-readable media as recited in claim 1, wherein
10 the loading the trusted core comprises copying at least a portion of the trusted core
11 from a remote device into the memory.

12
13 12. One or more computer-readable media as recited in claim 1, wherein
14 the loading the trusted core comprises copying at least a portion of the trusted core
15 from a chip of the computer.

16
17 13. One or more computer-readable media as recited in claim 1, wherein
18 the preventing comprises ignoring all requests for access to the memory from the
19 one or more central processing units and one or more bus masters.

20
21 14. One or more computer-readable media as recited in claim 1, wherein
22 the plurality of instructions further cause the one or more processors to perform
23 acts including:

24 extracting a cryptographic measure of the trusted core in the memory; and
25 storing the extracted cryptographic measure.

1
2 **15.** One or more computer-readable media as recited in claim 14,
3 wherein the plurality of instructions further cause the one or more processors to
4 perform acts including:

5 resetting a cryptographic processor;
6 requesting the cryptographic processor to extract the cryptographic
7 measure; and
8 receiving the extracted cryptographic measure from the cryptographic
9 processor.

10
11 **16.** One or more computer-readable media as recited in claim 1, wherein
12 the resetting each of the one or more central processing units comprises asserting a
13 processor bus reset signal to each of the one or more central processing units.

14
15 **17.** One or more computer-readable media as recited in claim 1, wherein
16 the plurality of instructions further cause the one or more processors to perform
17 acts including:

18 mapping a central processing unit reset vector to an initialization vector;
19 receiving a read request corresponding to the central processing unit reset
20 vector from the one central processing unit;
21 returning, in response to the read request, the initialization vector to the one
22 central processing unit; and
23 allowing the one central processing unit to access the memory beginning
24 with the initialization vector.
25

1 **18.** One or more computer-readable media as recited in claim 17,
2 wherein the initialization vector is an address within the trusted code in the
3 memory.

4
5 **19.** One or more computer-readable media as recited in claim 17,
6 wherein the plurality of instructions further cause the one or more processors to
7 perform acts including:

8 re-mapping the central processing unit reset vector to an additional central
9 processing unit start vector after returning the initialization vector to the one
10 central processing unit; and

11 returning, in response to any other read request corresponding to the central
12 processing unit reset vector from another central processing unit, the additional
13 central processing unit start vector.

14
15 **20.** One or more computer-readable media as recited in claim 19,
16 wherein the initialization vector is an address within the trusted code in the
17 memory and wherein the additional central processing unit start vector and the
18 initialization vector are different addresses within the trusted code in the memory.

19
20 **21.** One or more computer-readable media as recited in claim 19,
21 wherein both the initialization vector and the additional central processing unit
22 start vector are obtained from the trusted core.

1 22. One or more computer-readable media as recited in claim 1, wherein
2 the plurality of instructions further cause the one or more processors to perform
3 acts including loading microcode from the trusted core in memory into the one
4 central processing unit after resetting the central processing unit.

5
6 ~~23.~~ A method comprising:
7 booting, based on untrustworthy code, a computer;
8 loading a trusted core into memory; and
9 initiating secure execution of the trusted core.

10
11 24. A method as recited in claim 23, further comprising:
12 allowing execution of the trusted core to terminate; and
13 re-initiating secure execution of the trusted core without re-booting the
14 computer.

15
16 25. A method as recited in claim 23, further comprising:
17 allowing execution of the trusted core to terminate;
18 loading another trusted core into memory; and
19 initiating secure execution of the other trusted core.

20
21 26. A method as recited in claim 25, wherein the trusted core and the
22 other trusted core are different versions of the same trusted core.
23
24
25

1 **27.** A method as recited in claim 23, wherein the initiating comprises
2 initiating secure execution of the trusted core in response to an initialize trusted
3 core command received from one of the one or more central processing units.

4
5 **28.** A method as recited in claim 23, wherein the initiating comprises
6 initiating secure execution of the trusted core without requiring any additional bus
7 transactions to be supported by processors in the computer.

8
9 **29.** A method as recited in claim 23, wherein the initiating secure
10 execution of the trusted core comprises:

11 preventing each of one or more central processing units in the computer
12 from accessing the memory;

13 preventing each of one or more bus masters in the computer from accessing
14 the memory;

15 resetting each of the one or more central processing units;

16 allowing one central processing unit to access the memory and execute a
17 trusted core initialization process; and

18 after execution of the trusted core initialization process, allowing any other
19 central processing units and any of the one or more bus masters to access the
20 memory.

21
22 **30.** A method as recited in claim 29, further comprising:

23 mapping a central processing unit reset vector to an initialization vector;

24 receiving a read request corresponding to the central processing unit reset
25 vector from the one central processing unit;

1 returning, in response to the read request, the initialization vector to the one
2 central processing unit; and

3 allowing the one central processing unit to access the memory beginning
4 with the initialization vector.

5
6 **31.** A method as recited in claim 30, wherein the initialization vector is
7 an address within the trusted code in the memory.

8
9 **32.** A method as recited in claim 30, further comprising:
10 re-mapping the central processing unit reset vector to an additional central
11 processing unit start vector after returning the initialization vector to the one
12 central processing unit; and

13 returning, in response to any other read request corresponding to the central
14 processing unit reset vector from another central processing unit, the additional
15 central processing unit start vector.

16
17 **33.** A method as recited in claim 32, wherein the initialization vector is
18 an address within the trusted code in the memory and wherein the additional
19 central processing unit start vector and the initialization vector are different
20 addresses within the trusted code in the memory.

1 34. A method as recited in claim 23, wherein the loading the trusted
2 core comprises copying different portions of the trusted core from a plurality of
3 different sources including one or more of: a local mass storage device, a remote
4 device, and a local chipset.

5
6 35. One or more computer-readable memories containing a computer
7 program that is executable by a processor to perform the method recited in claim
8 23.

9
10 ~~36.~~ A method comprising:
11 allowing a computer to begin operation based on untrustworthy code;
12 loading, under the control of the untrustworthy code, additional code into
13 memory; and
14 initiating execution of the additional code in a secure manner despite the
15 untrustworthy code in the computer.

16
17 37. A method as recited in claim 36, wherein the initiating further
18 comprises initiating execution of the additional code in a secure manner despite
19 both the untrustworthy code in the computer and other pre-existent state of the
20 computer.

21
22 38. A method as recited in claim 36, wherein the initiating execution of
23 the additional code in a secure manner comprises:
24 preventing each of one or more central processing units in the computer
25 from accessing the memory;

1 preventing each of one or more bus masters in the computer from accessing
2 the memory;

3 resetting each of the one or more central processing units;

4 allowing one central processing unit to access the memory and execute a
5 code initialization process; and

6 after execution of the code initialization process, allowing any other central
7 processing units and any of the one or more bus masters to access the memory.

8
9 **39.** A method as recited in claim 36, wherein the initiating comprises
10 initiating execution of the additional code in a secure manner without requiring
11 any additional bus transactions to be supported by a processor in the computer.

12
13 **40.** A method as recited in claim 36, further comprising:
14 mapping a central processing unit reset vector to an initialization vector;
15 receiving a read request corresponding to the central processing unit reset
16 vector from the one central processing unit;

17 returning, in response to the read request, the initialization vector to the one
18 central processing unit; and

19 allowing the one central processing unit to access the memory beginning
20 with the initialization vector.

21
22 **41.** A method as recited in claim 40, further comprising:
23 re-mapping the central processing unit reset vector to an additional central
24 processing unit start vector after returning the initialization vector to the one
25 central processing unit; and

1 returning, in response to any other read request corresponding to the central
2 processing unit reset vector from another central processing unit, the additional
3 central processing unit start vector.

4
5
6 **42.** A method as recited in claim 36, further comprising:
7 remapping the trusted core to appear at an address where a central
8 processing unit starts executing after being reset.

9
10 **43.** A method as recited in claim 36, further comprising:
11 receiving, from a central processing unit, a read request corresponding to a
12 central processing unit reset vector;
13 responding to the read request with instructions to cause the central
14 processing unit to jump to a starting location of the trusted core.

15
16 **44.** A method as recited in claim 36, wherein the loading the additional
17 code comprises copying different portions of the additional code from a plurality
18 of different sources including one or more of: a local mass storage device, a
19 remote device, and a local chipset.

20
21 **45.** One or more computer-readable memories containing a computer
22 program that is executable by a processor to perform the method recited in claim
23 36.

1 **46.** A memory controller comprising:
2 a first interface to allow communication with a processor;
3 a second interface to allow communication with a system memory; and
4 a controller, coupled to the first interface and the second interface, to reset a
5 processor and to allow the processor to execute a code initialization process while
6 preventing any other processors from accessing the system memory.

7
8 **47.** A memory controller as recited in claim 46, wherein the memory
9 controller is included in a processor.

10
11 **48.** A memory controller as recited in claim 46, wherein the first
12 interface comprises a processor bus interface.

13
14 **49.** A memory controller as recited in claim 48, wherein the memory
15 controller operates without requiring the processor bus interface to support any
16 additional commands on the processor bus.

17
18 **50.** A memory controller as recited in claim 46, wherein the system
19 memory comprises a dynamic random access memory.

20
21 **51.** A memory controller as recited in claim 46, wherein the controller is
22 further to allow the processor to execute the code initialization process while
23 preventing any bus masters from accessing the system memory.

1 **52.** A memory controller as recited in claim 46, wherein the controller is
2 further to:

3 reset any other processor coupled to the memory controller prior to
4 allowing the processor to execute the code initialization process;

5 prevent any other processor and any bus master coupled to the memory
6 controller from accessing the system memory until the one process executes the
7 code initialization process; and

8 after execution of the code initialization process, allow any other central
9 processing units coupled to the memory controller and any bus masters coupled to
10 the memory controller to access the memory.

11
12 **53.** A memory controller as recited in claim 46, wherein the controller is
13 further to:

14 map a processor reset vector to an initialization vector;

15 receive a read request corresponding to the processor reset vector from the
16 processor;

17 return, in response to the read request, the initialization vector to the
18 processor; and

19 allow the processor to access the memory beginning with the initialization
20 vector.

21
22 **54.** A memory controller as recited in claim 53, wherein the
23 initialization vector is an address within the code initialization process.
24
25

1 **55.** A memory controller as recited in claim 53, wherein the controller is
2 further to:

3 re-map the processor reset vector to an additional processor start vector
4 after returning the initialization vector to the processor; and

5 return, in response to any other read request corresponding to the processor
6 reset vector from another processor, the additional processor start vector.

7
8 **56.** A memory controller as recited in claim 55, wherein the
9 initialization vector is an address within the code initialization process and
10 wherein the additional processor start vector and the initialization vector are
11 different addresses within the code initialization process.

12
13 **57.** An apparatus comprising:
14 a processor reset portion to assert a reset signal to a processor; and
15 a memory protector portion to prevent any bus master from accessing
16 memory until the processor completes execution of a trusted core initialization
17 process.

18
19 **58.** An apparatus as recited in claim 57, wherein the apparatus
20 comprises a programmable logic device.

21
22 **59.** An apparatus as recited in claim 57, wherein the processor reset
23 portion comprises a processor bus interface.

1 60. An apparatus as recited in claim 57, wherein the memory protector
2 portion comprises a control logic that ignores any request to access the memory
3 received from any bus master.

4
5 61. An apparatus as recited in claim 57, further comprising a controller,
6 coupled to the memory protector portion, to prevent another processor from
7 accessing memory until the processor completes execution of the trusted core
8 initialization process.

9
10 62. An apparatus as recited in claim 57, further comprising a controller,
11 coupled to the memory protector portion, to:

12 map a processor reset vector to an initialization vector;

13 receive a read request corresponding to the processor reset vector from the
14 processor;

15 return, in response to the read request, the initialization vector to the
16 processor; and

17 allow the processor to access the memory beginning with the initialization
18 vector.

19
20 63. An apparatus as recited in claim 62, wherein the controller is further
21 to:

22 re-map the processor reset vector to an additional processor start vector
23 after returning the initialization vector to the processor; and

24 return, in response to another read request corresponding to the processor
25 reset vector from another processor, the additional processor start vector.

1
2 64. An apparatus as recited in claim 57, further comprising a storage
3 portion in which a portion of the trusted core is stored.

4
5 65. An apparatus as recited in claim 64, wherein the portion of the
6 trusted core stored in the storage portion comprises a platform trusted core portion.

7
8 ~~66.~~ A computer comprising:
9 a processor;
10 a bus master;
11 a system memory; and
12 a memory controller coupled to the processor, the bus master, and the
13 system memory, the memory controller being configured to,
14 allow access to the system memory from the processor and the bus
15 master operating based on untrustworthy code,
16 reset the processor to begin a trusted core initialization process, and
17 prevent the bus master from accessing the system memory until after
18 the trusted core initialization process is completed.

19
20 67. A computer as recited in claim 66, further comprising a plurality of
21 additional processors and preventing the plurality of additional processors from
22 accessing the system memory until after the trusted core initialization process is
23 completed.

1 68. A method comprising:

2 allowing execution of different trusted cores in a computer to be initiated
3 serially without requiring the computer to be re-booted.
4

5 69. A method as recited in claim 68, wherein the allowing further
6 comprises allowing execution of the different trusted cores to be initiated at
7 arbitrary times.
8

9 70. A method as recited in claim 68, wherein the different trusted cores
10 are different versions of the same trusted core.
11

12 add
A₁